

IN THE  
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): HUNDT et al.

Confirmation No.: 5331

Application No.: 09/901,363

Examiner: Rampuria, S.

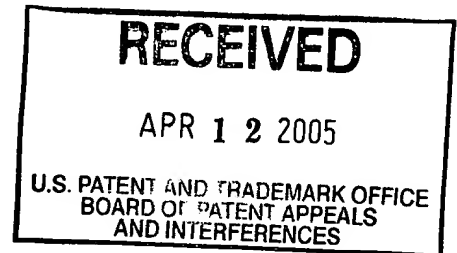
Filing Date: 07/09/2001

Group Art Unit: 2124

Title: OPTIMIZING AN EXECUTABLE COMPUTER PROGRAM HAVING LINKAGE SUPPORT  
FUNCTIONS

Mail Stop Appeal Brief-Patents  
Commissioner For Patents  
PO Box 1450  
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF



Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 02/08/2005.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( ) one month	\$120.00
( ) two months	\$450.00
( ) three months	\$1020.00
( ) four months	\$1590.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of \$500.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:  
Commissioner for Patents, Alexandria, VA  
22313-1450. Date of Deposit: 04/07/2005

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number \_\_\_\_\_ on \_\_\_\_\_

Number of pages:

Typed Name: Rennae Johnson

Signature: Rennae Johnson

Respectfully submitted,

HUNDT et al.

By LeRoy D. Maunu

LeRoy D. Maunu

Attorney/Agent for Applicant(s)

Reg. No. **35,274**

Date: **04/07/2005**


Telephone No.: **(651) 686-6633**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

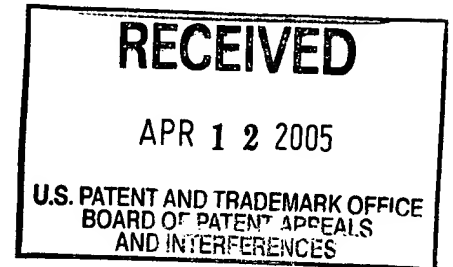
Appellant:	HUNDT et al.	Examiner:	Rampuria, S.
Serial No.:	09/901,363	Group Art Unit:	2124
Filed:	July 9, 2001	Docket No.:	10012768-1
Title:	OPTIMIZING AN EXECUTABLE COMPUTER PROGRAM HAVING LINKAGE SUPPORT FUNCTIONS		

---

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence and the papers, as described hereinabove, are being deposited in the United States Postal Service, as first class mail, in an envelope addressed to: Board of Patent Appeals and Interferences, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450, on April 7, 2005.

By   
Rennae Johnson

Board of Patent Appeals and Interferences  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450



Sir:

This is an Appeal Brief submitted pursuant to 37 C.F.R. § 41.37 for the above-referenced patent application.

**I. Real Party in Interest**

The real party in interest is Hewlett-Packard Company having a place of business at 1501 Page Mill Road, Palo Alto, CA. The above referenced patent application is assigned to Hewlett-Packard Company.

**II. Related Appeals and Interferences**

Appellant is unaware of any related appeals, interferences or judicial proceedings.

**III. Status of Claims**

Claims 1-15 are presented for appeal. The claims presented for appeal are in the attached Appendix of Appealed Claims.

**IV. Status of Amendments**

No amendments were filed subsequent to the final Office Action dated November 15, 2004.

## **V. Summary of Invention**

The invention provides a computer-implemented method for optimizing an executable program. "Linkage stub functions" are used in a program to call a function in an external library, and calls to linkage stub functions introduce overhead into program execution (p. 2, l. 15-23). The program includes a plurality of functions, and at least one of the functions has a first name associated with executable code that implements the function at a first address. The program also includes at least one linkage stub code segment that has code that branches to the first address (FIG. 1A, #104; p. 4, l. 19-24) and a symbolic name by which the function is invoked in the program (FIG. 1A, #114; p. 5, l. 1-5). The method comprises identifying branch instructions having target addresses that reference the linkage stub code segment (FIG. 1A, #114, 116; FIG. 2, #312; FIG. 4, #410; p. 5, l. 1-5; p. 6, l. 6-16; p. 7, l. 1 – p. 8, l. 6); and replacing the target addresses of the branch instructions with the first address (FIG. 1B, #122; FIG. 2, #314; FIG. 3, #412; p. 5, l. 6-12; p. 7, l. 5-8; p. 8, l. 6-12).

## **VI. Grounds of Rejection**

- A. Claims 1, 2, and 15 are rejected under 35 U.S.C. § 102(e) by "Coutant" (U.S. Patent No. 6,665,671 to Coutant *et al.*).
- B. Claims 3-5 are rejected under 35 U.S.C. § 103(a) over Coutant in view of admitted prior art.
- C. Claims 6, 7, and 11 are rejected under 35 U.S.C. § 103(a) over Coutant in view of "Koizumi" (U.S. Publication No. 2002/0026633 to Koizumi *et al.*).
- D. Claims 8-10 and 12-14 are rejected under 35 U.S.C. § 103(a) over Coutant and Koizumi in view of admitted prior art.

## **VII. Argument**

- A. **The rejection of claims 1, 2, 15 under 35 USC §102(e) is improper because the Examiner has failed show that Coutant teaches all the limitations of the claims.**

Claims 1 and 15

The Examiner has failed to show that Coutant teaches all the limitations of claims 1 and 15. Therefore, Applicants respectfully request that the rejection be reversed.

Claim 1 sets forth a method of optimizing an executable program having a plurality of functions. At least one of the functions has a first name associated with executable code that implements the function at a first address, and at least one linkage stub code segment has code that branches to the first address and a symbolic name by which the function is invoked in the program. The method includes identifying branch instructions having target addresses that reference the linkage stub code segment; and replacing the target addresses of the branch instructions with the first address. The Examiner clearly fails to show that these limitations are taught by Coutant.

The Examiner is mistaken in citing various sections of Coutant as teaching the limitations of claim 1 because Coutant's system is directed to optimizing access to shared data, not optimizing program code with function calls as claimed. Coutant's Background and Summary make clear that Coutant's approach deals with optimizing shared data access, not optimizing function calls. For example, Coutant's Background describes both remote code access, including function calls (col. 1, ll. 33-48) and remote data access (col. 1, ll. 49-67). These are clearly two different concepts. Coutant's Summary goes on to explain that his invention is for providing a method for optimizing and efficiently accessing shared data (col. 2, ll. 14-36). As the cited sections of Coutant clearly indicate and explained below, Coutant's approach is for remote data access, not optimizing function calls.

The Examiner mistakenly alleges that Coutant's col. 3, ll. 18-20 teach the limitations of identifying branch instructions having target addresses that reference the linkage stub code segment. The alleged teaching does not correspond to the claim limitations because this section of Coutant appears to only describe a remote function call. There is no teaching or suggestion that Coutant's process in any way identifies the branch instructions in a program as claimed.

The Examiner is also mistaken in the alleged correspondence of Coutant's col. 8, ll. 25-26 to the limitations of replacing the target addresses of the branch instructions with the first address. These teachings of Coutant do not correspond to the claim limitations because the entire paragraph makes clear that *load* instructions

are examined, not *branch* instructions (col. 8, ll. 8-28). Coutant replaces a linkage table *load* instruction with a *noop*, and an *indirect load* instruction is replaced with a *direct load* instruction. Not only does this teaching fail to suggest *branch* instructions, but the teaching says nothing of the claimed replacing of a target address. Therefore, the Examiner has failed to show that Coutant anticipates claim 1.

The Examiner similarly fails to show that Claim 15 is anticipated by Coutant.

Having failed to show that Coutant teaches all the limitations of claims 1 and 15, the rejection of claims 1 and 15 under §102(e) should be reversed.

#### Claim 2

The rejection of claim 2 fails for the same general reasons as set forth above in the response to the rejection of claim 1. That is, the limitations of claim 2 involve the target address of a branch instruction, not Coutant's linkage tables and load instructions.

Furthermore, claim 2 includes limitations of replacing the target address of the branch instructions with the first address only in functions that are reached during program execution. The Examiner cites Coutant's teachings at col. 8, l. 23-26 as corresponding to these limitations. However, there is no apparent reference in these teachings of replacing a target address of a branch instruction only in a function that is reached during program execution. There is no apparent suggestion in the cited teachings that Coutant's deals differently with those functions reached versus those functions not reached during program execution. Therefore, the Examiner has not shown that Coutant teaches all the limitations of claim 2.

Reversal of the rejection of claim 2 under 35 USC §102(e) is respectfully requested since Coutant is not shown to teach all the limitations of claim 2.

**B The rejection of claims 3-5 over Coutant in view of admitted prior art is improper because. because the Examiner has failed to establish a *prima facie* case of obviousness.**

The Examiner has failed to establish a *prima facie* case of obviousness for claims 3-5 under 35 USC §103(a) over Coutant in view of admitted prior art. Therefore, reversal of the rejection is respectfully requested.

Coutant should not be used as prior art under §103, and the rejections should be withdrawn. The Examiner uses Coutant as a basis in rejecting claims 3-5 (and also 6-14) under 35 U.S.C. §103(a). However, it is respectfully submitted that under 35 U.S.C. §103(c), Coutant should be removed as a reference because at the time of the present invention (with an application filing date of July 9, 2001), the present invention and Coutant were both owned, or subject to assignment to the same person. Therefore, Coutant is not eligible as prior art under §103, and the rejections should be withdrawn. The Examiner continues to apply Coutant under §103(a) even though the Amendment dated July 20, 2004 explained the Coutant should not be used to preclude patentability under §103(a).

Even though the Coutant-based rejections under §103 should be withdrawn based on §103(c), further arguments are set forth below to make clear that the Examiner has not established *prima facie* obviousness even if Coutant were not disqualified under §103(c).

The Examiner has failed to establish a *prima facie* case of obviousness because all the limitations have not been shown to be suggested by the references and a proper motivation for modifying the teachings of Coutant has not been provided.

The Examiner has not shown that Coutant suggests the limitations of claim 3 that include searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name; and replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored. The cited sections of Coutant involve *load* instructions, not *branch* instructions, and there is no apparent suggestion of any searching for a derivation of a first name as claimed. Furthermore, the cited teachings of Coutant involve replacing a linkage table load with a *no-op* instruction. Those skilled in the art will recognize the lack of correspondence between a *no-op* instruction and the replacement of an address in a branch instruction. There is no apparent correspondence to the claimed replacing a target address of a branch instruction with another address as claimed.

Therefore, the Examiner has not shown that Coutant suggests the limitations of claims 3-5 that further refine the limitations of claim 1 that are related to branch instructions.

The alleged motivation for modifying Coutant is improper because it simply restates that derivative names are known. There is no prior art evidence presented to indicate any desirability for searching a symbol table for a derivation of a name. The alleged motivation states that "it would have been obvious ... to incorporate the method of derivative or having underscore in the name as taught in prior art into the method of optimizing the executable program code as taught by Coutant ... because one of ordinary skill in the art would be motivated have underscore or derivative of the name at the time of execution to differentiate the function name as suggested in prior art (Applicant's specification, page 2, lines 17-19)." The alleged motivation provides no evidence that Coutant's approach would be improved if a symbol table were searched for a derivation of a function name. Furthermore, the alleged motivation does not provide any evidence to support modifying Coutant from optimizing shared data accesses to optimizing branches to functions through stubs. The alleged motivation is conclusory and based on hindsight, and therefore, improper.

The rejection of claims 3-5 over Coutant and admitted prior art should be reversed because the Examiner has failed to show all the limitations are suggested by the combination and has failed to provide a proper motivation for combining the references.

**C. The rejection of claims 6, 7 and 11 over Coutant in view of Koizumi is improper because the Examiner has failed to establish a *prima facie* case of obviousness.**

The Examiner has failed to establish a *prima facie* case of obviousness for claims 6, 7, and 11 under 35 USC §103(a) over Coutant in view of Koizumi. Furthermore, as explained above, Coutant should not be used as prior art. Therefore, reversal of the rejection is respectfully requested.

The further arguments set forth below make clear that the Office Action does not establish *prima facie* obviousness even if Coutant were not disqualified under §103(c). The Examiner has failed to establish a *prima facie* case of obviousness because all the limitations of claims 6, 7, and 11 have not been shown to be suggested by the references and a proper motivation for modifying the teachings of Coutant with teachings of Koizumi has not been provided.

### Claims 6 and 7

Claim 6 includes further limitations of replacing function entry points in the executable program with breakpoints, whereby breakpointed functions are generated; and upon encountering a breakpoint of a breakpointed function during program execution, identifying within the breakpointed function branch instructions that target linkage stub functions. The Examiner is in error in relying on various sections of Koizumi as suggesting these limitations.

For example, the Examiner cites Koizumi's page 17, para. 387 as suggesting these limitations. However, this section describes setting a flag to indicate a breakpoint at a program statement. This is not suggestive of actually replacing the function entry points as claimed. Furthermore, the cited para. 389 has no apparent relevance to the limitations of identifying instructions in a breakpointed function that target linkage stub functions. Para. 389 describes processing of an execute command. The described processing involves extracting the address of an instruction, for example, from a program counter, and checking whether a breakpoint is set at the determined address. This is clearly not suggestive of identifying instructions within a breakpointed function that target linkage stub functions (also note that neither Coutant nor Koizumi suggest or refer to "instructions that target linkage stub functions"). Thus, the limitations of claim 6 are not shown to be suggested by the Coutant-Koizumi combination.

Claim 7 depends from claim 7, and the limitations are not shown to be suggested for at least the reasons set forth above for claim 6.

The alleged motivation for modifying Coutant with the cited teachings of Koizumi is improper because it is conclusory and based on hindsight. The alleged motivation states that "it would have been obvious ... to incorporate the method of setting breakpoint as taught by Koizumi into the method of optimizing the executable program code as taught by Coutant ... because one of ordinary skill in the art would be motivated to set a break point within the code so that the translation of the code is appropriate for the target machine as suggested by Koizumi ...". No evidence is provided to indicate that code translation is a suitable modification to Coutant's system. Coutant's system apparently discusses only the optimization of shared data access (Abstract). Supplanting Koizumi's setting of a breakpoint flag and determining of breakpoints by way of getting the current instruction address for purposes of code translation do not appear to have any particular relevance to



Coutant's modifying of load instructions for optimizing remote data access. The alleged motivation is conclusory and based on hindsight, and therefore, improper.

Reversal of the rejection of claims 6 and 7 is respectfully requested because the Examiner has failed to show all the limitations are taught by the Coutant-Koizumi combination and has failed to provide a proper motivation for making the combination.

#### Claim 11

The rejection of claim 11 should also be reversed because the Examiner has failed to show that the Coutant-Koizumi combination suggests all the claim limitations and has failed to provide a proper motivation for combining Koizumi with Coutant.

Claim 11 includes limitations of replacing entry points of linkage stub code segments in the executable program with breakpoints, whereby breakpointed linkage stubs are generated; and upon encountering a breakpoint of a breakpointed linkage stub during program execution, changing a target address of a branch instruction that branched to the breakpointed linkage stub to reference the function referenced by the breakpointed linkage stub. The same teachings of Koizumi alleged to correspond to the limitations of claim 7 are also alleged to correspond to the limitations of claim 11.

As explained above in regards to claim 7, Koizumi's teachings are not shown to suggest any of the claimed replacing of entry points with breakpoints. Thus, Koizumi's teachings are also not shown to suggest the claim 11 limitations of replacing entry points of linkage stub code segments with breakpoints.

The alleged motivation for combining Koizumi with Coutant is improper as set forth above in regards to claim 7.

The rejection claim 11 over the Coutant-Koizumi combination should be reversed because the Examiner has failed to show all the limitations are suggested by the combination and failed to provide a proper motivation for combining the references.

**D. The rejection of claims 8-10 and 12-14 over Coutant and Koizumi in view of admitted prior art is improper because the Examiner has failed to establish a *prima facie* case of obviousness.**

The rejection of claims 8 - 10 and 12 - 14 under 35 U.S.C. 103(a) as being unpatentable over the Coutant-Koizumi combination in view of admitted prior art should be reversed. *Prima facie* obviousness is not established because all the limitations are not shown by the Office Action to be suggested by the combination, the alleged motivation is improper, and no reasonable likelihood of successfully making the combination has been shown.

Claims 8 and 12 include the limitations of claim 3, claims 9 and 13 include the limitations of claim 4, and claims 10 and 14 include the limitations of claim 5. The Examiner has failed to show that the limitations of claims 8-10 and 12-14 are suggested by the prior art for at least the reasons set forth above in section B.

The alleged motivation for making the Coutant-Koizumi-admitted-prior-art combination is deficient for the reasons set forth above in regards to making the Coutant-Koizumi combination and the reasons set forth above in regards to making the Coutant-admitted-prior-art combination.


Therefore, the rejection of claims 8-10 and 12-14 over the Coutant-Koizumi-admitted-prior-art combination should be reversed.

#### **VIII. Conclusion**

In view of the above, Appellant submits that the rejections are improper, the claimed invention is patentable, and that the rejections of claims 1-15 should be reversed. Appellant respectfully requests reversal of the rejections as applied to the appealed claims and allowance of the entire application.

Respectfully submitted,

CRAWFORD MAUNU PLLC  
1270 Northland Drive – Suite 390  
St. Paul, MN 55120  
(651) 686-6633

By:   
Name: LeRoy D. Maunu  
Reg. No. 35,274

## APPENDIX OF APPEALED CLAIMS

1. A computer-implemented method for optimizing an executable program having a plurality of functions and at least one function with a first name associated with executable code that implements the function at a first address and at least one linkage stub code segment having code that branches to the first address and a symbolic name by which the function is invoked in the program, comprising:
  - identifying branch instructions having target addresses that reference the linkage stub code segment; and
  - replacing the target addresses of the branch instructions with the first address.
2. The method of claim 1, further comprising replacing the target address of the branch instructions with the first address only in functions that are reached during program execution.
3. The method of claim 1, further comprising:
  - searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name; and
  - replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.
4. The method of claim 1, further comprising:
  - searching a symbol table for an entry having a symbolic name that matches the first name with an underscore prefix and reading a linkage stub address associated with the symbolic name; and
  - replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.
5. The method of claim 1, further comprising:

searching a symbol table for an entry having a symbolic name that matches the first name with an underscore suffix and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

6. The method of claim 1, further comprising:

replacing function entry points in the executable program with breakpoints, whereby breakpointed functions are generated; and

upon encountering a breakpoint of a breakpointed function during program execution, identifying within the breakpointed function branch instructions that target linkage stub functions.

7. The method of claim 6, further comprising:

storing original instructions from the function entry points prior to replacement with the breakpoints;

upon encountering a breakpoint of a breakpointed function during program execution, restoring the original instruction to the entry point of the breakpointed function.

8. The method of claim 6, further comprising:

searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

9. The method of claim 6, further comprising:

searching a symbol table for an entry having a symbolic name that matches the first name with an underscore prefix and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

10. The method of claim 6, further comprising:

searching a symbol table for an entry having a symbolic name that matches the first name with an underscore suffix and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

11. The method of claim 1, further comprising:

replacing entry points of linkage stub code segments in the executable program with breakpoints, whereby breakpointed linkage stubs are generated; and

upon encountering a breakpoint of a breakpointed linkage stub during program execution, changing a target address of a branch instruction that branched to the breakpointed linkage stub to reference the function referenced by the breakpointed linkage stub.

12. The method of claim 11, further comprising:

searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

13. The method of claim 11, further comprising:

searching a symbol table for an entry having a symbolic name that matches the first name with an underscore prefix and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

14. The method of claim 11, further comprising:

searching a symbol table for an entry having a symbolic name that matches the first name with an underscore suffix and reading a linkage stub address associated with the symbolic name; and

replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored.

15. An apparatus for optimizing an executable program having a plurality of functions and at least one function with a first name associated with executable code that implements the function at a first address and at least one linkage stub code segment having code that branches to the first address and a symbolic name by which the function is invoked in the program, comprising:

means for identifying branch instructions having target addresses that reference the linkage stub code segment; and

means for replacing the target addresses of the branch instructions with the first address.

## EVIDENCE APPENDIX

None.

## RELATED PROCEEDINGS APPENDIX

None.



Receipt is hereby acknowledged for the following in the U.S. Patent and Trademark Office:

Applicant: HUNDT ET AL.

For: OPTIMIZING AN EXECUTABLE COMPUTER PROGRAM  
HAVING LINKAGE SUPPORT FUNCTIONS

Docket No.: 10012768-1 (HPCO.049PA)

Serial No.: 09/901,363

Date of Deposit: April 7, 2005

- ☒ Transmittal sheet containing Certificate under 37 CFR 1.8.
- ☒ Appeal Brief (15 pages).
- ☒ 2 Return Postcards

Patent



David M. Mason / rpa  
Hewlett-Packard Company  
Legal Department - MS 4059  
11000 Wolfe Road  
Cupertino, CA 95014-0691



CRAWFORD MAUNU PLLC  
1270 NORTHLAND DRIVE  
SUITE 390  
MENDOTA HEIGHTS, MN 55120

© 2002 recycled

Receipt is hereby acknowledged for the following in the U.S. Patent and Trademark Office:

Applicant: HUNDT ET AL.

For: OPTIMIZING AN EXECUTABLE COMPUTER PROGRAM  
HAVING LINKAGE SUPPORT FUNCTIONS

Docket No.: 10012768-1 (HPCO.049PA)

Serial No.: 09/901,363

Date of Deposit: April 7, 2005

- ☒ Transmittal sheet containing Certificate under 37 CFR 1.8.
- ☒ Appeal Brief (15 pages).
- ☒ 2 Return Postcards

Patent